Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

NormalF-Net: Normal Filtering Neural Network for Feature-preserving Mesh Denoising

Zhiqi Li [a], Yingkui Zhang [b], Yidan Feng [a], Xingyu Xie [c], Qiong Wang [b,*], Mingqiang Wei [a,*], Pheng-Ann Heng [d]

[a] *School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China*
[b] *Shenzhen Key Laboratory of Virtual Reality and Human Interaction Technology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China*
[c] *School of Electronics Engineering and Computer Science, Peking University, China*
[d] *Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong*

ARTICLE INFO

Article history:
Received 26 March 2020
Accepted 16 April 2020

Keywords:
NormalF-Net
Normal filtering
Neural network
Low-rank matrix recovery

ABSTRACT

Normal filtering is a fundamental step of feature-preserving mesh denoising. Methods based on convolutional neural networks (CNNs) have recently made their debut for normal filtering. However, they require complicated voxelization and/or projection operations for regularization, and afford an overall denoising accuracy with few powers of preserving surface features. We devise a novel normal filtering neural network algorithm, which we call as NormalF-Net. NormalF-Net consists of two cascaded subnetworks with a comprehensive loss function. The first subnetwork learns mapping from non-local patch-group normal matrices (NPNMs) to their ground-truth low-rank counterparts for denoising, and the second subnetwork learns mapping from the recovered NPNMs to the ground-truth normals for normal refinement. Different from existing learning-based methods, NormalF-Net, which bridges the connection between CNNs and *geometry domain knowledge* of *non-local similarity*, can not only preserve surface features when removing different levels and types of noise, but be free of voxelization/projection. NormalF-Net has been validated on different datasets of meshes with multi-scale features yet corrupted by noise of different distributions. Experimental results consistently demonstrate clear improvements of our method over the state-of-the-arts in both noise-robustness and feature awareness.

1. Introduction

3D scanning and sensing techniques have become widely applied to capture surface mesh data of physical objects. Recent advances in these techniques have allowed breakthroughs in a variety of application domains, such as reverse engineering [1–3], machine vision guided tasks [4,5], biomedical engineering [6–9], and cultural heritage preservation [10]. Although the 3D scanning technology constantly improves, these devices generate abundant noisy data that hinder subsequent geometry processing tasks [11,12]. These challenging issues highlight the need for computationally inexpensive, parameter-free and high-fidelity approaches for mesh denoising.

Mesh denoising, formulated as $\mathbf{M}^* = \mathbf{M} + \varepsilon$, aims to recover the ground-truth $\mathbf{M}$ from its observation $\mathbf{M}^*$ corrupted by noise ε. However, it is an ill-posed problem, since the clean mesh and noise are all unknown in advance [13,14]. This is why there is a significant interest in mesh denoising capable of improving the quality of mesh models to make them maximally compliant with their intended uses.

The key to success of mesh denoising is to remove noise while preserving surface features as accurately as possible. In the literature, there are rich works on this topic, such as filter-based [15,16, 21–23], optimization-based [18,24,25], and learning-based methods [13,20,26,27]. Filter-based ones, which are mainly inherited from ideas of image processing, utilize sophisticatedly-designed regularities to preserve/recover surface features but with tedious parameters adjustment, and they could not deal with meshes with large-scale noise well. Optimization-based ones, which best fit the noisy input and one or more priors of shape (e.g., piecewise flat) and/or noise (e.g., Gaussian noise along the mesh vertex normal), may not be generalized to really-scanned meshes with unseen shape and noise patterns. Currently, learning-based methods have made their debut for mesh denoising based on normal filtering. After learning mapping from the noisy inputs to their ground-truth counterparts in an offline stage, they can be

* Corresponding authors.
E-mail addresses: wangqiong@siat.ac.cn (Q. Wang), mqwei@nuaa.edu.cn (M. Wei).

https://doi.org/10.1016/j.cad.2020.102861
0010-4485/© 2020 Elsevier Ltd. All rights reserved.

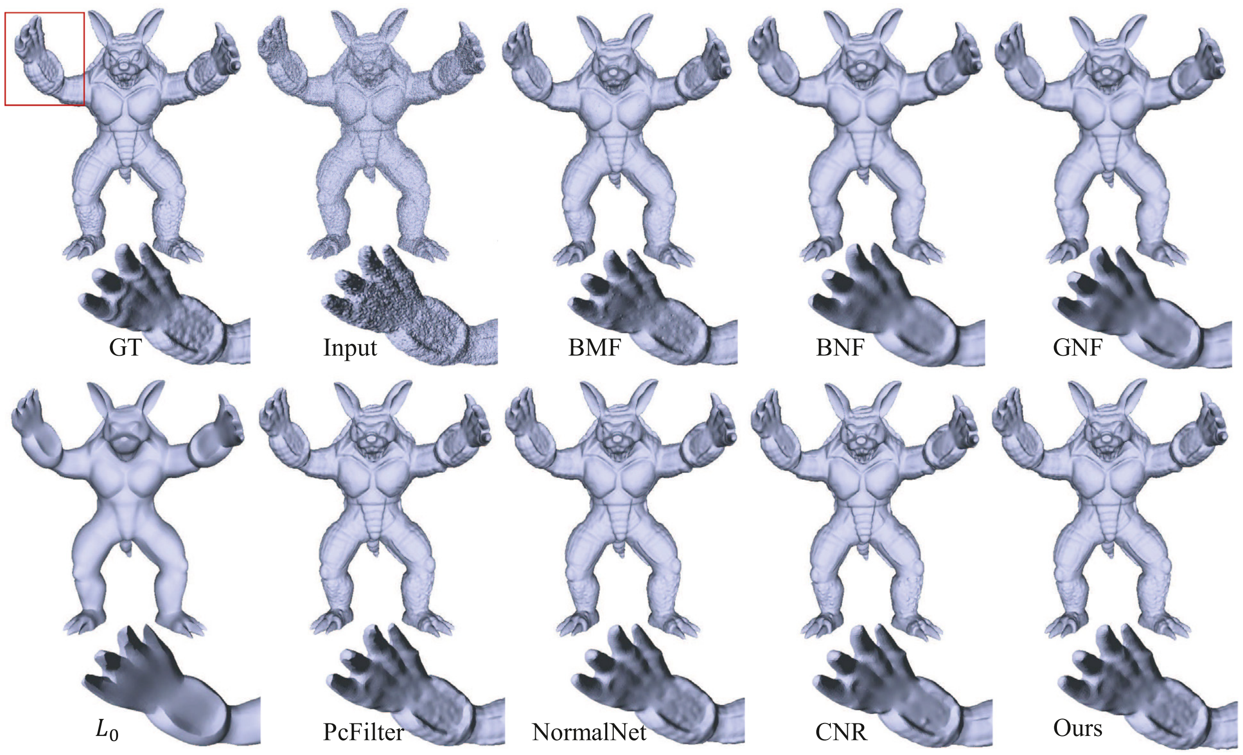


Fig. 1. Comparison of denoising approaches on the Armadillo mesh added by Gaussian noise ($\sigma_E = 0.2$). From the 1st row to the 2nd row: The ground truth, the noisy mesh, the denoised results of Fleishman et al.'s BMF [15] with parameters ($n = 10$), Zheng et al.'s BNF [16] ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$), Zhang et al.'s GNF [17] ($\sigma_r = 0.35, n_1 = 20, n_2 = 10$), He et al.'s L_0 minimization [18] ($\mu = \sqrt{2}, \alpha = 0.1\overline{\gamma}, \lambda = 0.02\ell_e^2\overline{\gamma}$), Wei et al.'s PcFilter [19] ($\sigma_s = 0.4, n_1 = 2, n_2 = 10$), Zhao et al.'s NormalNet [20], Wang et al.'s CNR [13], and our NormalF-Net.

automatically executed on new cases sharing similar geometry and noise characteristics of the trained models in a runtime stage. However, learning-based methods often involve complicated voxelization and/or projection operations on the spatial and topological irregular meshes for being compatible with current CNN architectures. Moreover, we also show existing learning-based methods solely pursue an overall denoising accuracy with the loss of surface features to a certain extent.

Before the rise of CNNs in mesh denoising, *geometry domain knowledge* (e.g., piecewise smoothness of the underlying surface, sparseness of surface sharp features, and similarity of surface patches, etc.) has been exploited in mesh filters, which are quite helpful in removing noise while preserving surface features. Although combining CNNs with geometry domain knowledge is a promising direction for feature-preserving mesh denoising, (1) geometry domain knowledge is difficult to be integrated into the input of CNNs directly; and (2) these traditional filters cannot tune parameters automatically to produce optimal denoising results by learning from data.

More local structures (patches) than one with similar geometry exist on a surface mesh, which is considered by existing works [19,28] as geometry domain knowledge of non-local similarity. These similar structures (represented by normals in each structure/patch) can be grouped together to form a non-local patch-group normal matrix (NPNM). Such NPNM should be low-rank due to high linear correlation among these similar patches, but actually possesses a high rank due to noise. Different from existing non-local similarity-based mesh denoising techniques which involve tedious parameters tuning and often lead to over-fitting, we do not intend to devise a low-rank matrix recovery model, but propose an automatical and non-local low-rank normal filtering neural network algorithm for feature-preserving mesh denoising, which is called NormalF-Net. Specifically, we select similar patches and pack their normals to form an NPNM. By simple operations, the column's elements (lined up by a patch's normals) in NPNM are well arranged without disorder and invariant to rotation. Therefore, we can first develop a subnetwork for denoising by learning mapping from noisy NPNMs to their ground-truth low-rank counterparts, and then develop another subnetwork for normal refinement by learning from the recovered NPNMs to their ground-truth normals being processed.

Although we follow the practical two-stage framework of normal filtering and vertex updating, NormalF-Net is different from existing mesh denoising approaches in four aspects: (1) the runtime denoising stage of NormalF-Net is completely automatic; (2) some learning-based methods have to apply voxelization/projection strategies to convert the irregular local structures of meshes [20] to structured formats for feeding into CNNs [29], while our method can avoid such voxelization/projection operations; (3) the first subnetwork of NormalF-Net mimics the low-rank matrix recovery, whereas, such geometry domain knowledge of non-local similarity is never integrated into learning-based denoising methods; and (4) the single network is insufficient to remove noise and preserve surface features simultaneously, while NormalF-Net, which absorbs the advantages of both the traditional and learning-based methods, adopt dual networks for denoising and normal refinement.

Our method shows clear improvements over the state-of-the-arts in terms of noise-robustness and feature awareness, as shown in Fig. 1. To the best of our knowledge, there are no works that have designed such a denoising-and-normal-refinement neural network by using NPNMs. Overall, the contributions of the proposed approach can be summarized in the following points:

- We propose a cascaded normal filtering neural network algorithm, called NormalF-Net, which the first subnetwork is responsible for denoising and the second subnetwork refines normals for better feature preservation.

• Based on the geometry domain knowledge of non-local similarity, we construct a series of non-local patch-group normal matrices (NPNMs) which can be easily fed into the network without any complicated voxelization/projection operation. Meanwhile, other learning-based geometric processing tasks may benefit from our well-formatted NPNMs.

2. Related work

The main issue in mesh denoising is to remove noise while preserving surface features. A variety of denoising-related methods have been proposed in recent years. We can roughly divide them into three categories: filter-based, optimization-based, and data-driven methods.

2.1. Filter-based methods

This type can be further divided into vertex-based and face-based filtering schemes. The pioneering vertex-based filtering schemes, such as Fleishman et al. [15] and Jones et al. [30], utilize bilateral filters to adjust vertex positions directly.

A large number of works have focused on the normal filtering framework to remove noise, aiming at developing more effective strategies that carry out the smoothing through computing local statistics on the normals, such as mean and median filtering [31], alpha-trimming [32], fuzzy median [33], and bilateral normal filtering [16]. Face-based filtering schemes first perform filtering on face normals and then updating the positions of vertices to match the filtered face normals accordingly.

To improve the feature-preserving capability of bilateral filtering, Zhang et al. [17] successfully apply joint bilateral filtering, in which the averaged normal of a most consistent local patch is selected as the guidance information. In spite of that, the construction of the guidance normal fails to adapt to complex features, leading to unsatisfactory denoising performance around these features. Subsequently, Li et al. [34] estimate guidance normals by the corner-aware neighborhood, which compensates the limitation in [17]. Assume that the normal change satisfies piecewise constant, Wei et al. [35] propose to cluster faces into piecewise smooth patches, and filter normals by leveraging complementary information from both the vertex and face normal fields. Another set of methods first detect the geometry features from the noisy input via various techniques including quadric fitting [36], element-based normal voting tensor [37], normal variance clustering [35] and iterative graph-cut [38] and then apply different filters that are adaptive to feature and non-feature parts. Although the aforementioned strategies can recover strong features, the shallow features are still difficult to be detected that may lead to over-smoothing.

2.2. Optimization-based methods

This type formulates denoising as a geometrical optimization problem, which is based on the input mesh information and a set of constraints defined by the priors of the ground-truth geometry and noise distribution.

Sparse optimization. Based on the observation that sharp features may be sparse on 3D models, the position and normal of vertex on the mesh can be combined to jointly measure the smoothness of the local area and the sparsity of the geometric features, so that the noise and features can be distinguished accurately. For example, He et al. [18] employ an L_0 norm to minimize curvatures of a surface, except with sharp features. Wang et al. [24] and Wu et al. [39] perform L_1 optimization to recover sharp features. Zhang et al. [25] combine total variation and piecewise constant function space for variational mesh denoising. For these methods, it is often critical to correctly estimate the differential geometry. Moreover, solving a global optimization is typically at expensive time cost.

Nonlocal similarity and low-rank matrix recovery. Nonlocal similarity is a powerful prior of natural images for image denoising [40–43]. They find from robust statics that similar patches exist on an image, which can be collaborated for handling high-level artifacts in images. Nonlocal similarity has also applied in 3D surface processing that produce promising denoising results [19,44]. In total, the normals in similar patches are used to form a matrix; then, the normal estimation problem is regarded as a low-rank matrix approximation problem, aiming at removing different scales of noise while preserving surface features.

2.3. Data-driven methods

With the increasing popularity of CNNs, researchers in graphics have attempted to employ deep neural networks for 3D model processing. For instance, Wang et al. [13] propose a cascaded learning model by mapping the filtered facet normal descriptor (FND) extracted from the neighborhood of a noisy mesh facet to the noise-free mesh facet normal, and use the modeled function to compute new facet normals. Wang et al. [26] design a two-step normal variation learning method by utilizing reverse normal filter, where the second step compensates for the loss of geometry in the first learning step. Thus, the geometry features can be better preserved. Zhao et al. [20] propose the Normal-Net to embed deep network into guided normal filtering (GNF), which applies a voxelization strategy on each face to transform the irregular local structure into the regular volumetric representation.

Notably, some interesting denoising works on point cloud also deserve our attention. PointNet [45] is one of the first network architectures that can handle point cloud data. Subsequently, Roveri et al. [46] propose point projection layer for converting unordered points to regularly sampled height maps. Similar to [46], Chen et al. [47] define a rotation-invariant height-map patch (HMP) for each point and each HMP encodes the field of projection distances from the surrounding points to the filtered tangent plane of the target point. Besides, in [48], a discretized Hough space representing normal directions is projected onto a structure amenable to deep learning.

3. Overview

Mesh denoising usually involves the estimation of two conflicting components of the noisy surface: noise (to be removed) and geometric details (to be preserved). By assuming that both noise and geometric details to be high frequency and their magnitudes to be similar to each other, it is challenging for a single network to learn both components simultaneously. Even though going deeper with plain neural networks cannot always lead to better feature-preserving mesh denoising performance. This motivates us to design a dual convolutional neural network. NormalF-Net synergizes to denoise and refine the lost geometric details caused by denoising.

Non-local similarity has been statically validated in noise removal from images and surfaces with the steps of: (1) A local window/patch is associated with the referred image pixel or mesh facet. (2) Similar windows/patches of a noisy image/mesh are grouped and reshaped as vectors (assembling pixel intensities or facet normals as the vector's elements). (3) A non-local patch-group matrix can be constructed by stacking these similar patch-vectors as the matrix's columns. (4) Due to its columns being highly linearly correlated, this patch-group matrix should be low-rank but currently possesses a high rank because of noise.

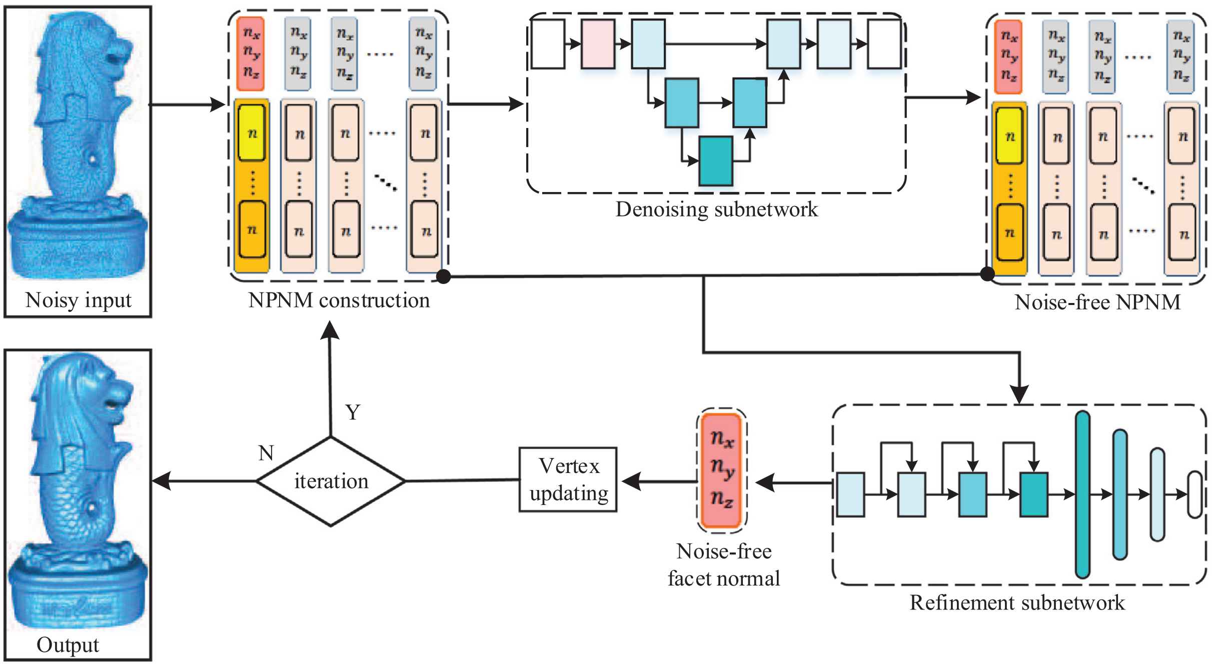


Fig. 2. NormalF-Net consists of the construction of non-local patch-group normal matrix (NPNM), the dual networks for denoising and normal refinement respectively, and the operation of vertex updating. It iteratively learns mapping between the noisy geometry and the ground-truth geometry from a training dataset, and utilizes the learned mappings for feature-preserving mesh denoising.

As a result, a low-rank matrix recovery model will be devised to solve such a convex optimization problem by the traditional denoising methods.

Instead of devising a low-rank matrix recovery model which fine-tunes its parameters to avoid surface over-fitting, we mimic the procedure of low-rank matrix recovery by learning between the noisy geometry of non-local patches and the clean geometry of their ground-truth counterparts for denoising. One more merit is that we do not need any voxelization/projection operation, since the non-local patch-group matrix has regularly encoded the local patch geometry for feeding into current neural network architectures.

After denoising, we learn between the recovered patch-group normal matrix and the corresponding clean facet normal for normal refinement. At the top level, such a denoising-and-normal-refinement neural network by using NPNMs is shown in Fig. 2.

4. NormalF-Net

4.1. NPNM construction

The main challenge of applying CNNs to mesh geometry processing comes from the irregular structures of meshes. Inspired by the non-local low-rank matrix recovery, we construct a set of regular NPNMs that can be utilized not only for denoising but also for flexibly feeding neural networks. In detail, each patch P (consisting of a given number of mesh facets, say N_P) around a facet is reshaped as a patch-vector, which forms a column of NPNM, and the $S_k - 1$ patches with most similar geometry, as the other columns of NPNM, can be grouped together. Since the normal vector is three-dimensional, the size of NPNM is $N_P \times S_k \times 3$.

4.1.1. Patch grouping

In 3D meshes, a patch P simply denotes the centered facet plus its $N_P - 1$ neighboring facets (the size of such a patch is N_P). The degree of similarity between these patches can be measured by the robust normal tensor voting technique.

Tensor Voting: The normal voting tensor for a mesh facet f_i is formulated as the sum of normal covariance matrices from both f_i and its neighboring facets:

$$T(f_i) = \sum_{f_j \in P} \mu_j \mathbf{n}_j \mathbf{n}_j^{\mathrm{T}}, \tag{1}$$

where P is the associated patch of the facet f_i, μ_j is a weighted coefficient defined by [49], and $\mathbf{n}_j$ is the normal of f_j in P; $T(\cdot)$ is symmetric positive semi-definite and can be decomposed as

$$T(f_i) = \lambda_1 \mathbf{e}_1 \mathbf{e}_1^T + \lambda_2 \mathbf{e}_2 \mathbf{e}_2^T + \lambda_3 \mathbf{e}_3 \mathbf{e}_3^T, \tag{2}$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ are its eigenvalues, and $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ are the corresponding unit eigenvectors.

We utilize the eigenvalues to measure the geometry similarity among the patches as [19]

$$\rho_{i,j} = \|\lambda_{1,i} - \lambda_{1,j}\|_2^2 + \|\lambda_{2,i} - \lambda_{2,j}\|_2^2 + \|\lambda_{3,i} - \lambda_{3,j}\|_2^2. \tag{3}$$

From the above metric, we could cluster $S_k - 1$ patches with the most geometry similarity to the patch associated with the reference facet.

Patch Size and the Number of Similar Patches: We empirically set the patch's size $N_P = 16$, and find the number of $S_k = 16$ of similar patches from the 8-ring neighboring facets of a reference facet.

4.1.2. Reshaping

Reshaping Order: The reference facet f_{ref} is placed at the first column as the starting facet. Then, one facet from the 1-ring neighboring facets of f_{ref} which is most similar to f_{ref} in terms of Eq. (3), is chosen as the seed facet. Therefore, the 1-ring neighboring facets can be stacked one by one from this seed facet in a counter-clockwise order. And we continue to apply the same operation on the 2-ring neighboring facets until a given number (i.e., N_P) of facets is obtained. Fig. 3 shows the whole pipeline of how to reshaping a patch and forming NPNM.

Invariant to Rotation: The facet normal orientations in one patch may be different from the other similar patches. For example, Fig. 4 shows a CAD-like model which possesses many

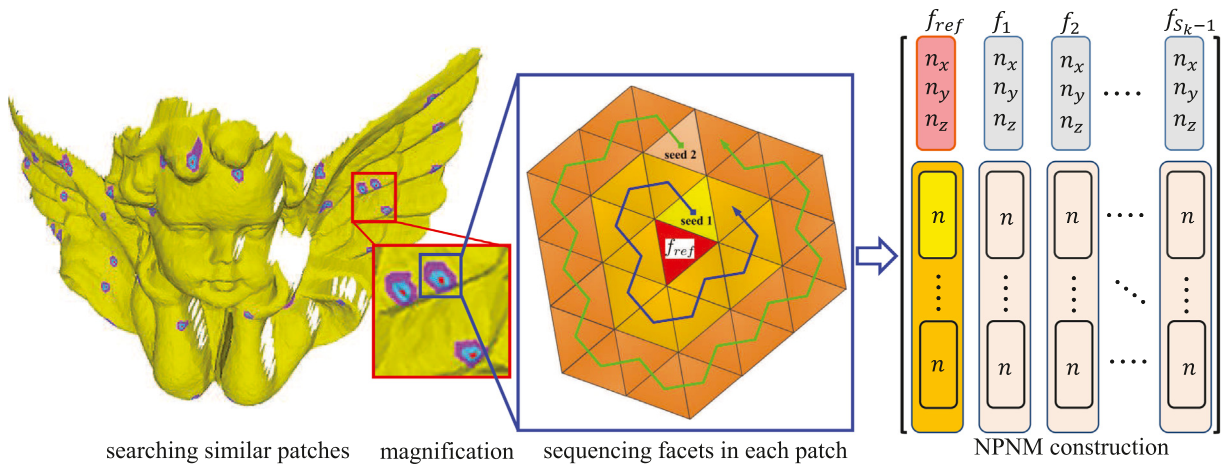


Fig. 3. Non-local patch-group normal matrix (NPNM) construction. Given a local patch with its center aligned with the reference (centered) facet f_{ref} being processed, we group patches with similar geometry by using the eigenvalues of normal tensor voting. Then, each patch is reshaped as a patch-vector by using the facet normal as its element from a seed facet to find other facets with a ring-by-ring scheme. Each patch-vector is invariant to global rigid transformation after a simple rotation operation. Finally, these patch-vectors are constructed as an NPNM, which is very suitable for feeding into the current neural network architectures.

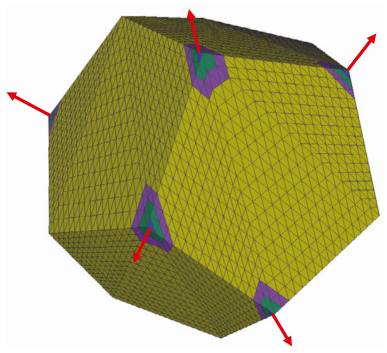


Fig. 4. The facet normal orientations from one patch (e.g., the facets around a corner) may be different from the other similar patches. To solve this problem, we use the result of normal voting tensor to make facet normals in each patch be invariant to rotation.

sharp corners, and one can see that the normal orientation of each central facet is inconsistent to other central facets from the corners. To make the facet normals in each patch be invariant to rotation, we introduce a rotation matrix, denoted as $\mathbf{R}$. Similar to [13], the rotation matrix consists of three eigenvectors from normal voting tensor in Eq. (2): $\mathbf{R} = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$. By multiplying each facet normal with the reverse matrix $\mathbf{R}^{-1}$, all of the normals in the patch will be aligned. Therefore, the constructed NPNMs are invariant to rotation.

Remark. We have enlarged the size of NPNM from 16 × 16 to 32 × 32. However, this improves the denoising results slightly but with a higher computational cost. Fig. 5 shows that there are very small differences between the two patch sizes on a 3D mesh. We also obtain similar results on other meshes. Therefore, we have decided to set the resolution of NPNM to be 16 × 16, which behaves well in our all experiments.

4.2. Network architecture

Our NormalF-Net consists of two cascaded sub-networks with a comprehensive loss function which co-operate to denoise and preserve surface features. In the following, we will introduce the denoising network and the refinement network, respectively.

4.2.1. Denoising subnetwork

The key idea of NormalF-Net is to mimic the procedure of low rank recovery by deep learning and further to refine the normal. Therefore, the denoising subnetwork accepts the noisy NPNMs and outputs the ground-truth counterparts. Considering that this matrix-to-matrix regression requires precise pixel level predictions, the proposed network contains several contracting blocks and expanding blocks, inspired by U-Net [50]. This design allows us to aggregate features from multiple dimensions, which preserves context information and enables precise locations. As illustrated in Fig. 7, all convolution layers are followed by the Relu function, except that the last layer applies Tanh to ensure the output within the range of a unit normal vector. Note that each normal in the recovered NPNMs (i.e., the output of denoising subnetwork) is normalized since the NPNM is actually a set of normal vectors. The loss function is written as

$$Loss_1 = \frac{1}{N_p \times S_k} \| \wedge (M_t)_3 - M_g \|_2^2 \tag{4}$$

where M_t and M_g are the recovered and the ground-truth NPNMs, respectively. $\wedge(\cdot)_i$ denotes the normalization along dimension i.

4.2.2. Refinement subnetwork

Since both noise and geometric details are high frequency and have similar magnitudes, part of the meshes suffer from over-smoothing when learning the denoised low-rank matrices from the noisy NPNMs in the denoising network. To alleviate this problem, we further propose a refinement subnetwork to learn the mapping from the recovered matrix M_t to the ground-truth normal. Benefiting from the powerful ResNet [51] to extract features and avoid gradient vanishing, we devise the refinement network as shown in Fig. 6. Specifically, we adopt three residual blocks to extract features and three fully connected layers to linearly transform high dimensional features into the feature space of the normal vectors. Tanh function is applied at the end of the architecture. In our implementation, we expand the input features by concatenating the original NPNMs, which compensates the information lost in the pixel level denoising.

Let us denotes N_t and N_g as the network output and the ground-truth normal, respectively. The objective function of our refinement subnetwork can be written as

$$Loss_2 = \| \wedge (N_t) - N_g \|_2^2. \tag{5}$$

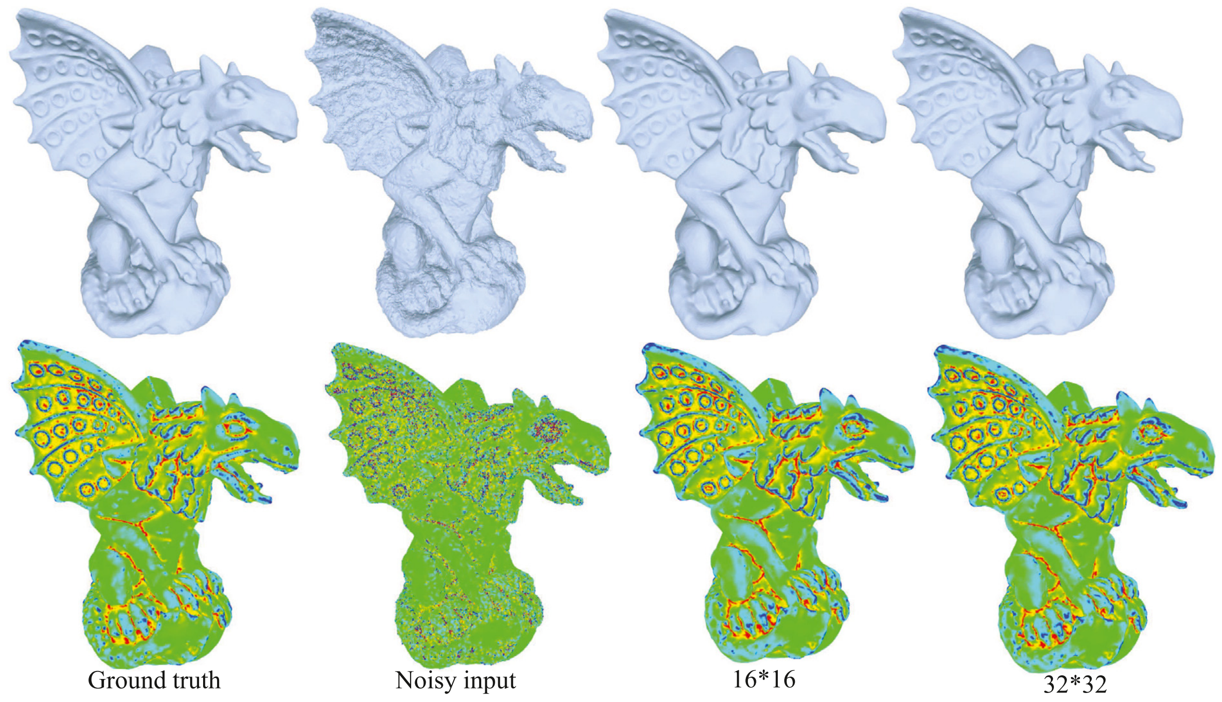


Fig. 5. The visual and numerical differences between the patch sizes of 16 × 16 and 32 × 32 are very slight. The first row shows the mesh geometry, and the second row shows the mean curvature visualization. The average angular differences to the ground-truth are 23.65° (random Gaussian noise of $\sigma_E = 0.2$), 6.21° (16 × 16), 6.19° (32 × 32).

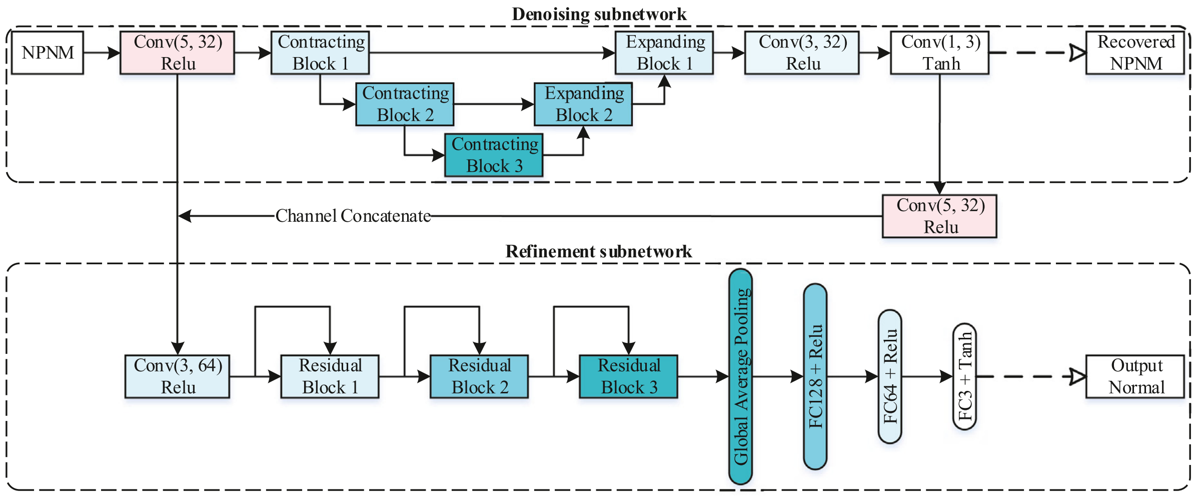


Fig. 6. The architecture of our neural network, which consists a denoising module (upper) to regress recovered NPNMs, and a refinement module (below) to output the final normals.

Loss Function: The comprehensive loss function of our cascaded two subnetworks can be formulated as

$$\mathbf{Loss} = \alpha Loss_1 + (1-\alpha)Loss_2, \tag{6}$$

where α is a non-negative trade-off weight. In our experiments, we have found that $\alpha = 0.7$ provides a good balance between patch denoising and normal refinement.

4.3. Vertex update

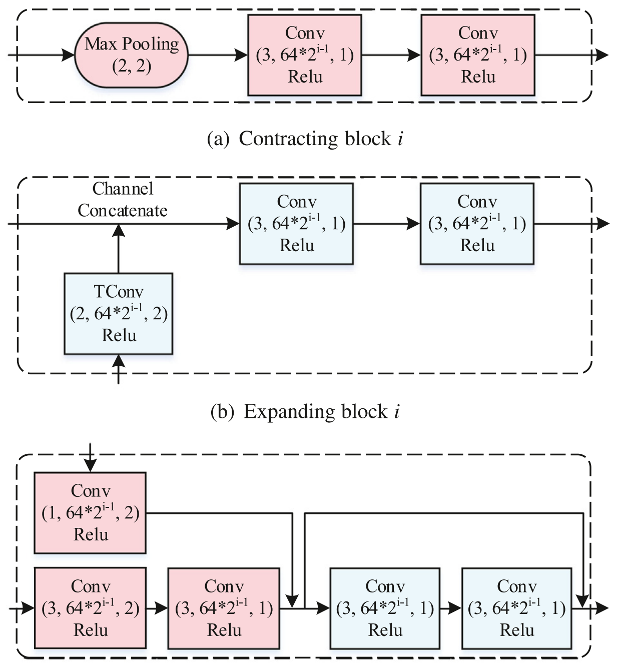
Based on previous obtained facet normals from the refinement subnetwork, the noisy mesh vertices are updated to match the set of new normals $\hat{\mathbf{n}}$. According to the iterative approach of [52], we update the vertices' positions as follows:

$$\mathbf{v}_i^{\text{new}} := \mathbf{v}_i^{\text{old}} + \frac{1}{3\left|\Omega\left(\mathbf{v}_i\right)\right|} \sum_{f_k \in \Omega(\mathbf{v}_i)} \sum_{\mathbf{e}_{ij} \in \partial f_k} \left(\hat{\mathbf{n}}_k \cdot \left(\mathbf{v}_j^{old} - \mathbf{v}_i^{old}\right)\right) \hat{\mathbf{n}}_k \tag{7}$$

where $\Omega\left(\mathbf{v}_i\right)$ is the one-ring neighborhood of v_i and e_{ij} is the edge of triangle f_k with end vertices v_i and v_j. This scheme is essentially a gradient descent method to minimize the semi-definite quadratic error function. We set the iteration number as 20 in our implementation.

5. Experiment and results

Competitors. To verify the competitive effect of NormalF-Net, we quantitatively and qualitatively compare it to different


Max Pooling (2, 2)
Conv (3, 64*2^{i-1}, 1) Relu
Conv (3, 64*2^{i-1}, 1) Relu
Channel Concatenate
Conv (3, 64*2^{i-1}, 1) Relu
Conv (3, 64*2^{i-1}, 1) Relu
TConv (2, 64*2^{i-1}, 2) Relu
Conv (1, 64*2^{i-1}, 2) Relu
Conv (3, 64*2^{i-1}, 2) Relu
Conv (3, 64*2^{i-1}, 1) Relu
Conv (3, 64*2^{i-1}, 1) Relu
Conv (3, 64*2^{i-1}, 1) Relu

(a) Contracting block *i*

(b) Expanding block *i*

(c) Residual block *i*

Fig. 7. The blocks used in our scheme. Max pooling (2,2) means the pooling size is 2 × 2 and the stride is 2, Conv(k, $64 * 2^{i-1}$, s) means the convolution uses a $k \times k$ filter, the channel number is $64 * 2^{i-1}$ and the stride is s. TConv means the transposed convolution for up-sampling. (a) Note that the first contracting block has no max-pooling layer since there is no down-sampling in the first block. (b) Expanding Block i has two inputs for feature aggregation, the one is from the same dimension and the other is from the transposed convolution output of a higher dimension. (c) Residual block i uses one input for feature extraction and another for down-sampling through 1 × 1 convolution. Similarly, the first residual block has no 1 × 1 convolution branch.

types of mesh denoising methods, such as the one-step method, i.e., bilateral mesh denoising (BMF) [15], the optimization-based method, i.e., L_0 minimization mesh denoising (L_0) [18], the filter-based methods, i.e., bilateral normal filtering (BNF) [16], and guided normal filtering (GNF) [17], the low-rank based method, i.e., PcFilter [19], and the learning-based methods, i.e., cascaded normal regression (CNR) [13], and Normal-Net [20]. The results of L_0, BMF, BNF, GNF and CNR are generated from the source codes released by their authors or implemented by a third party [17], while the results of Normal-Net and PcFilter are courtesy of the authors.

5.1. Dataset collection

NormalF-Net has been tested on the repository of CNR [13]. The repository contains four kinds of datasets that possess different types and levels of noise, as well as various surface features, i.e., synthetic, Kinect v1, Kinect v2, and Kinect-Fusion datasets.

For the synthetic dataset, there are 21 meshes for training, and 30 meshes for testing which are further divided into CAD-like models, smooth models, and models with rich features. All are added with fixed-scale Gaussian noise to the mesh vertices along the vertex normal.

For the real-world raw datasets, they come from Microsoft Kinect, which the noise is introduced by the devices. In detail, the three datasets are acquired by Microsoft Kinect v1, Microsoft Kinect v2, and the Kinect-Fusion technique. Moreover, the models obtained by a high-resolution laser scanner have been used as noise-free ground truths. The six physical objects to be scanned are called David, Boy, Girl, Big Girl, Cone, and Pyramid, respectively. Following CNR [13], we train our NormalF-Net on the aforementioned four datasets separately.

Table 1
Ablation study with the error E_a.

	CAD	Smooth	Feature
S1	4.5428	3.4502	5.6627
S2	4.5114	3.4427	5.6032
S3	4.3790	3.4407	5.4010

5.2. Ablation study

We perform an ablation study to demonstrate the effectiveness of the denoising-and-normal-refinement architecture of NormalF-Net. The whole algorithm of NormalF-Net is decomposed as

- S1: learning from noisy NPNMs to the ground-truth NPNMs, and using the median filter in [19] to update facet normals.
- S2: learning from noisy NPNMs to the ground-truth facet normals.
- S3: our NormalF-Net.

Table 1 shows the average angle differences of the three schemes. Therefore, we observe that the whole algorithm of NormalF-Net is more effective than the two decomposed schemes.

5.3. Comparisons

Synthetic Dataset. There are 63 noisy models, which have 3 million facets in total, and the corresponding 3 million NPNMs for training. Since there are constantly quite a few facets with similar local geometric features in a model, to reduce the network computation and save memory resources, we adopt a down-sampling operation on each model that participates in the training, and ensure that 500,000 NPNMs are inputed into the network. The testing dataset contains 90 models with different-level noise and multi-scale surface features in a total number of 7 million facets.

For training NormalF-Net, we choose the Adam algorithm with the default settings as the optimization method, the mini-batch size is 64 and the initial learning rate is 0.0001. The decay of learning rate occurs at every 400 training steps, for which the decay rate is 0.95. The training is implemented in Matlab with Deep Learning Toolbox. Our NormalF-Net is iteratively performed for three times and each iteration is trained for 5 epochs, which can yield satisfactory results.

In addition, we also evaluate the performance of each algorithm through following metrics:

- E_a: the average angle difference between the output facet normals and the ground-truth facet normals.
- E_d: the average one-sided Hausdorff distance from the mesh reconstructed by the output facet normals to the ground-truth mesh.

Our NormalF-Net can lead to better visual and numerical denoising results, which the different scales of surface features are well-preserved, as shown in Figs. 8, 9, 12, 14.

Denoise feature-rich models. The results of our NormalF-Net are more faithful to the ground-truths, Fig. 1 first shows the Armadillo model with varieties of repetitive details, sharp features, as well as smooth areas, and results by applying different methods. Compared with other methods, NormalF-Net recovers more structures (e.g., palm) and introduces no additional artifacts

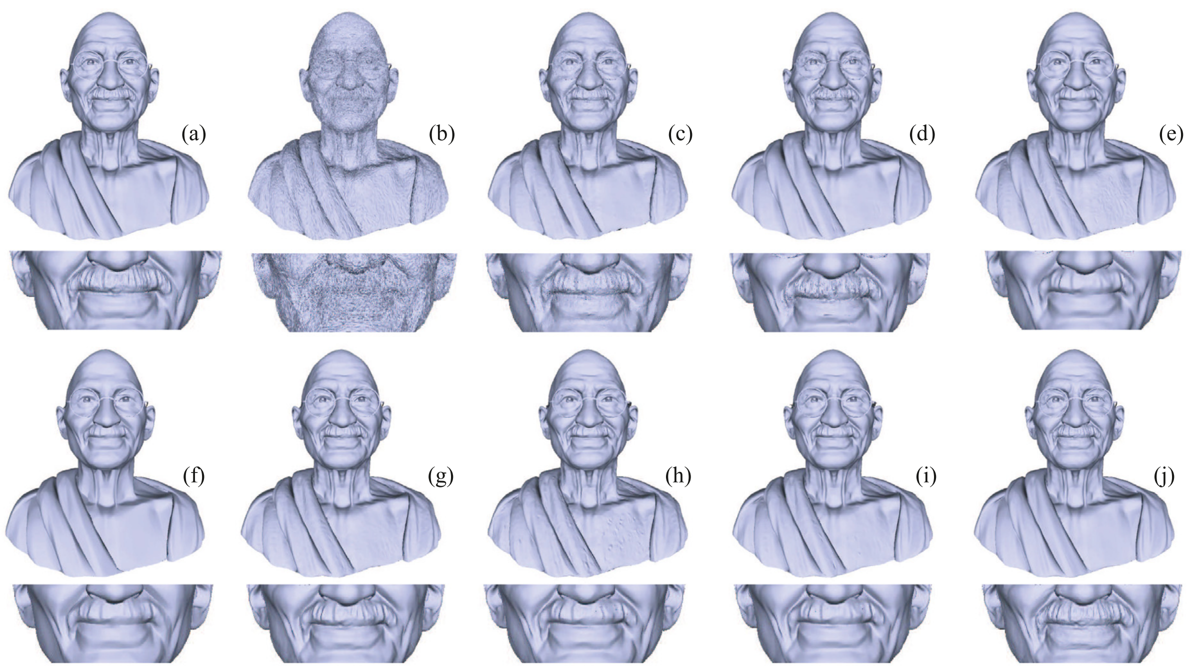


Fig. 8. On a noisy input mesh (b) with its ground-truth (a), we compare our NormalF-Net (j) with the state-of-the-art denoising methods: (c) BMF [15] ($n = 15$); (d) BNF [16] ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$); (e) GNF [17] ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$); (f) L_0 minimization [18] ($\mu = \sqrt{2}, \alpha = 0.1\overline{\gamma}, \lambda = 0.02\ell_e^2\overline{\gamma}$); (g) PcFilter [19] ($\sigma_s = 0.35, n_1 = 10, n_2 = 10$); (h) NormalNet [20]; (i) CNR [13]; and (j) our NormalF-Net. From the magnified fragments we can see, NormalF-Net could better recover the surface details than other denoising methods.

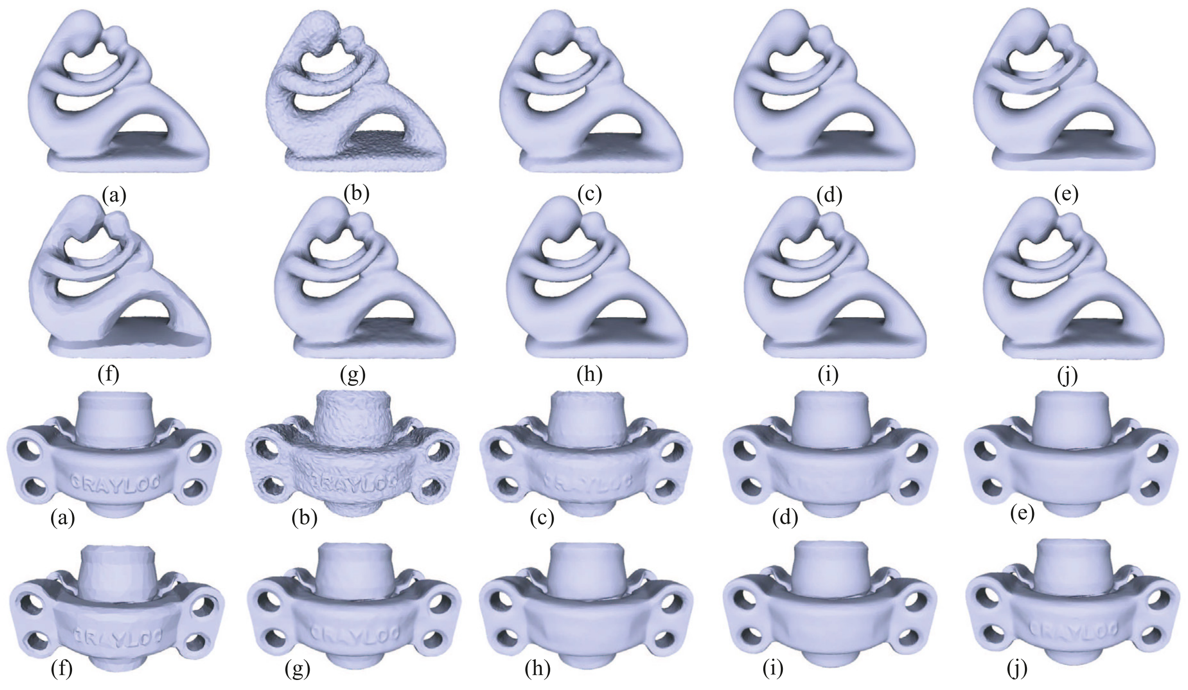


Fig. 9. Denoising of the Fertility model and the Grayloc model, from (a) to (j): (a) ground-truth, (b) the noisy models (the Fertility and the Grayloc with Gaussian noise of $\sigma_E = 0.1$), (c) BMF [15] ($n = 10$) ($n = 20$); (d) BNF [16] ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$) ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$); (e) GNF [17] ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$) ($\sigma_s = 0.35, n_1 = 20, n_2 = 20$); (f) L_0 minimization [18] ($\mu = \sqrt{2}, \alpha = 0.1\overline{\gamma}, \lambda = 0.02\ell_e^2\overline{\gamma}$); (g) PcFilter [19] ($\sigma_s = 0.3, n_1 = 2, n_2 = 8$) ($\sigma_s = 0.4, n_1 = 2, n_2 = 8$); (h) NormalNet [20]; (i) CNR [13]; and (j) our NormalF-Net.

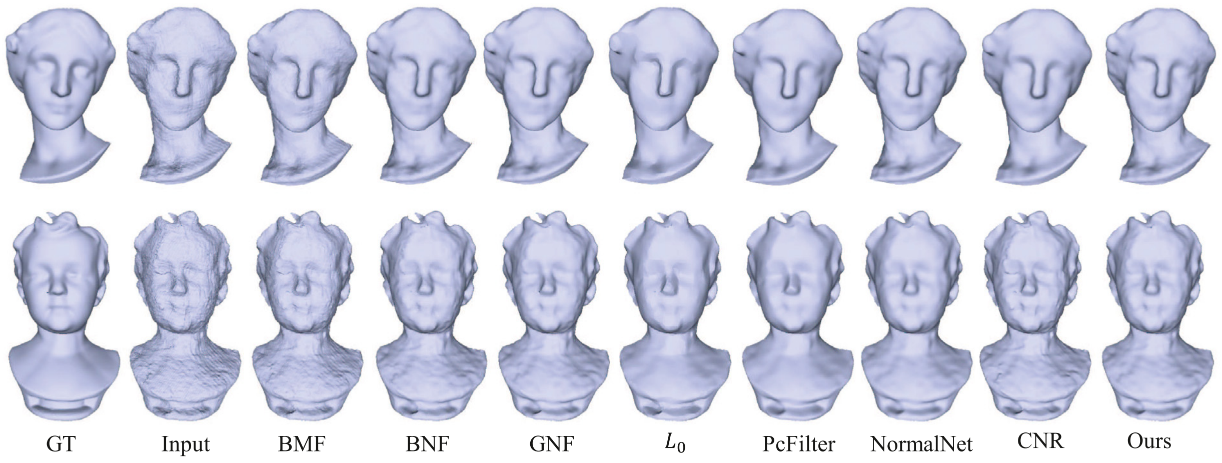


Fig. 10. Denoising of the Big Girl model and the Boy model (real scans), from left to right: The ground-truth, the noisy input, BMF [15] ($n = 25$) ($n = 25$), BNF [16] ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$) ($\sigma_s = 0.3, n_1 = 20, n_2 = 10$), GNF [17] ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$) ($\sigma_s = 0.4, n_1 = 20, n_2 = 20$), L_0 minimization [18] ($\mu = \sqrt{2}, \alpha = 0.1\overline{\gamma}, \lambda = 0.02\ell_e^2\overline{\gamma}$), PcFilter [19] ($\sigma_s = 0.4, n_1 = 10, n_2 = 10$) ($\sigma_s = 0.4, n_1 = 30, n_2 = 20$), NormalNet [20], CNR [13], and our NormalF-Net.

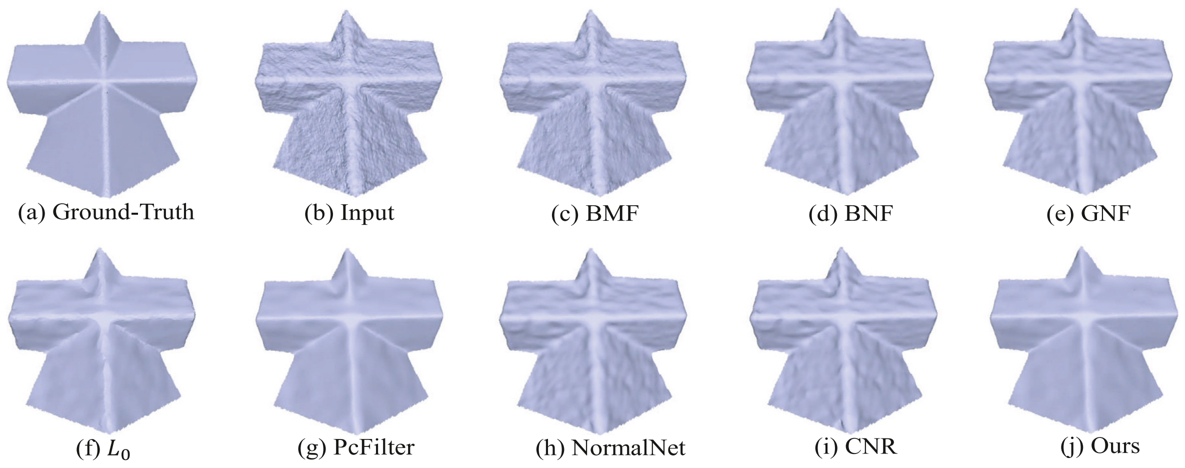


Fig. 11. Denoising of the Pyramid model: (a) ground-truth; (b) noisy input (c) BMF [15] ($n = 35$); (d) BNF [16] ($\sigma_s = 0.35, n_1 = 20, n_2 = 10$); (e) GNF [17] ($\sigma_s = 0.45, n_1 = 15, n_2 = 10$); (f) L_0 minimization [18] ($\mu = \sqrt{2}, \alpha = 0.1\overline{\gamma}, \lambda = 0.02\ell_e^2\overline{\gamma}$); (g) PcFilter [19] ($\sigma_s = 0.45, n_1 = 20, n_2 = 20$); (h) NormalNet [20]; (i) CNR [13]; and (j) our NormalF-Net.

on these multi-scale features (e.g., sharpening or blurring effects), while the filter-based methods and optimization-based method either over-sharpen these palms or over-smooth them. More examples can be found on Figs. 8 and 14, see particularly the beard of GandHi model, the eyes of Eros model (Fig. 14 1st row), and the hair of the Child model (Fig. 14 2nd row), where our method can better preserve details of these models, while others are not.

Denoise CAD models. Figs. 9 and 14 fourth-row show the denoised results of the noisy CAD models. For the noisy Fertility model and Grayloc model (corrupted by random Gaussian noise of $\sigma_E = 0.1$), our NormalF-Net recovers more small details than the others and has no over-blurring effect. For example, our method preserves the characters of the Grayloc model more clearly than the BNF, GNF, Normal-Net and CNR. For the Sharp-sphere model with many curved edges and sharp corners, which make it challengable to recover them exactly when removing noise, our method can easily remove such noise around sharp edges and corners and recover these features better. Note that, to make the comparison fair, we have fine-tuned the parameters of the other methods and choose their best results.

Real Scans. Apart from the synthetic data, our NormalF-Net also outperforms the state-of-the-arts in the challenging real-world raw data. We further train the network on the real scan datasets Kinect v1, Kinect v2 and Kinect-Fusion separately. Since the total number of facets for training is relatively small, the max-epochs is increased in the dataset to ensure that at least 30,000 training steps are performed. In detail, 5 training epochs are used for Kinect v1, Kinect v2, and 10 for Kinect-Fusion. The optimization method and parameters in training are the same as those in synthetic dataset. According to visual results in Figs. 10 and 11, our method consistently leads to better results of retaining surface smoothness and surface features while the others over-sharpen the features and/or have more bumps on smooth regions. It is also clearly showed in Fig. 13 that ours are more faithful to the ground-truth.

5.4. Computational cost

The proposed NormalF-Net is performed on a desktop PC with a 2.2 GHz Intel Xeon E5-2650 CPU, 64GB RAM and a NVIDIA GTX-1080Ti. The time performance of all the methods involved in the

Table 2
Timing (seconds) comparisons with the state-of-the-arts.

Model	SharpSphere (Fig. 14-4)	Fertility (Fig. 9-1)	Grayloc (Fig. 9-2)	Eros (Fig. 14-1)	Gargoyle (Fig. 5)
Faces	20 882	27 954	68 580	100 000	171 112
BMF [15]	0.64	0.75	2.55	2.64	7.55
BNF [16]	0.27	0.66	1.38	2.39	3.86
GNF [17]	3.45	4.81	14.77	25.49	51.76
L_0 [18]	15.09	27.72	72.75	162.78	158.81
PcFilter [19]	51.63	72.23	172.15	246.47	473.53
NormalNet [20]	836.12	1132.42	5418.27	9163.24	20763.28
CNR [13]	1.27	1.71	3.35	5.16	10.04
NormalF-Net	97.37	109.63	344.76	480.55	975.05

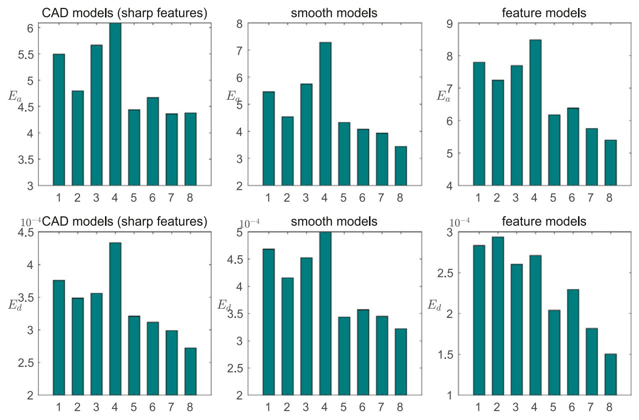


Fig. 12. Comparison with state-of-the-art methods (1–8 correspond to BMF, BNF, GNF, L_0, PcFilter, NormalNet, CNR, and ours). Bar chart in 1st row: the average normal angular difference E_a. Bar charts in 2nd row: the average distance E_d. Each column corresponds to one of the models. Higher bars are truncated for better illustration.

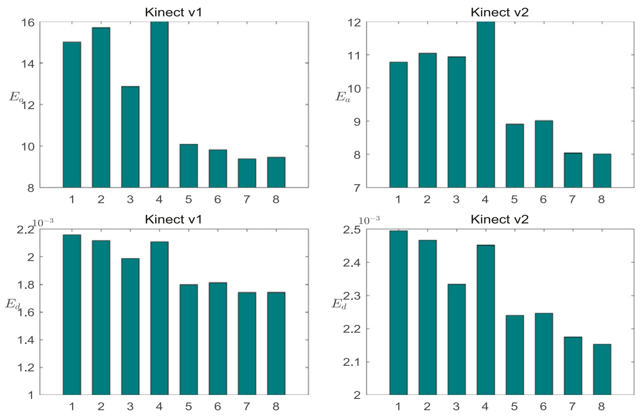


Fig. 13. Comparison with state-of-the-art methods on the *Kinect v1* (1st column) and *Kinect v2* (2nd column) datasets. Bar chart in 1st row: the average normal angular difference E_a. Bar charts in 2nd row: the average distance E_d. Higher bars are truncated for better illustration.

comparison is shown in Table 2. As we mentioned before, NPNM is free of complicated voxelization, which claim the superiority of our NormalF-Net over NormalNet [20]. However, compared with the filter-based methods BMF [15], BNF [16], GNF [17], and the learning-based CNR [13] that has been optimized to use the filter operator, our method is slightly inferior. We have also found that our NormalF-Net is more time-consuming than L_0 [18] and PcFilter [19].

6. Conclusion and future work

Deep learning, as an effective tool, has been largely employed in geometric processing tasks. However, few of recent approaches combines CNNs and geometry domain knowledge together, which we think it is a challenging but very promising trend. As one significant type of geometric processing tasks, the traditional wisdoms of mesh denoising have been developed for a long time, due to its ill-posedness. By now, few methods can serve as a mesh denoising panacea: they generate results with a tradeoff between noise removal and feature preservation.

We have proposed a novel normal filtering neural network algorithm, called NormalF-Net. NormalF-Net bridges the connection between CNNs and geometry domain knowledge of non-local similarity. Different form existing learning-based techniques, NormalF-Net does not require complicated voxelization and/or projection operations for regularization, but affords a powerful denoising ability with preserving surface features. However, we have also found that, (1) we could not bring our NormalF-Net's superiority into full play on CAD-like models, since they seldom possess geometry details except sharp edges and corners. All the methods like NormalNet, CNR, and BNF could well handle CAD-like models. (2) NormalF-Net has no obvious advantage in running time, we believe this relates to our current implementation. The construction of NPNM can be easily accelerated in parallel with GPU and octree space partition which further speeds up the retrieval of similar patches. In the future, we will fully consider the acceleration of NormalF-Net, and make it suitable for mobile phones as an application tool. We will also employ the feature descriptor NPNM for other geometry related tasks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the reviewers for their worthwhile comments. This work was supported by the Key-Area Research and Development Program of Guangdong Province, China (No. 2019B010149002), the National Natural Science Foundation of China (No. 61502137), the China Postdoctoral Science Foundation (No. 2016M592047), and the Shenzhen Science and Technology Program, China (No. JCYJ20170413162256793).

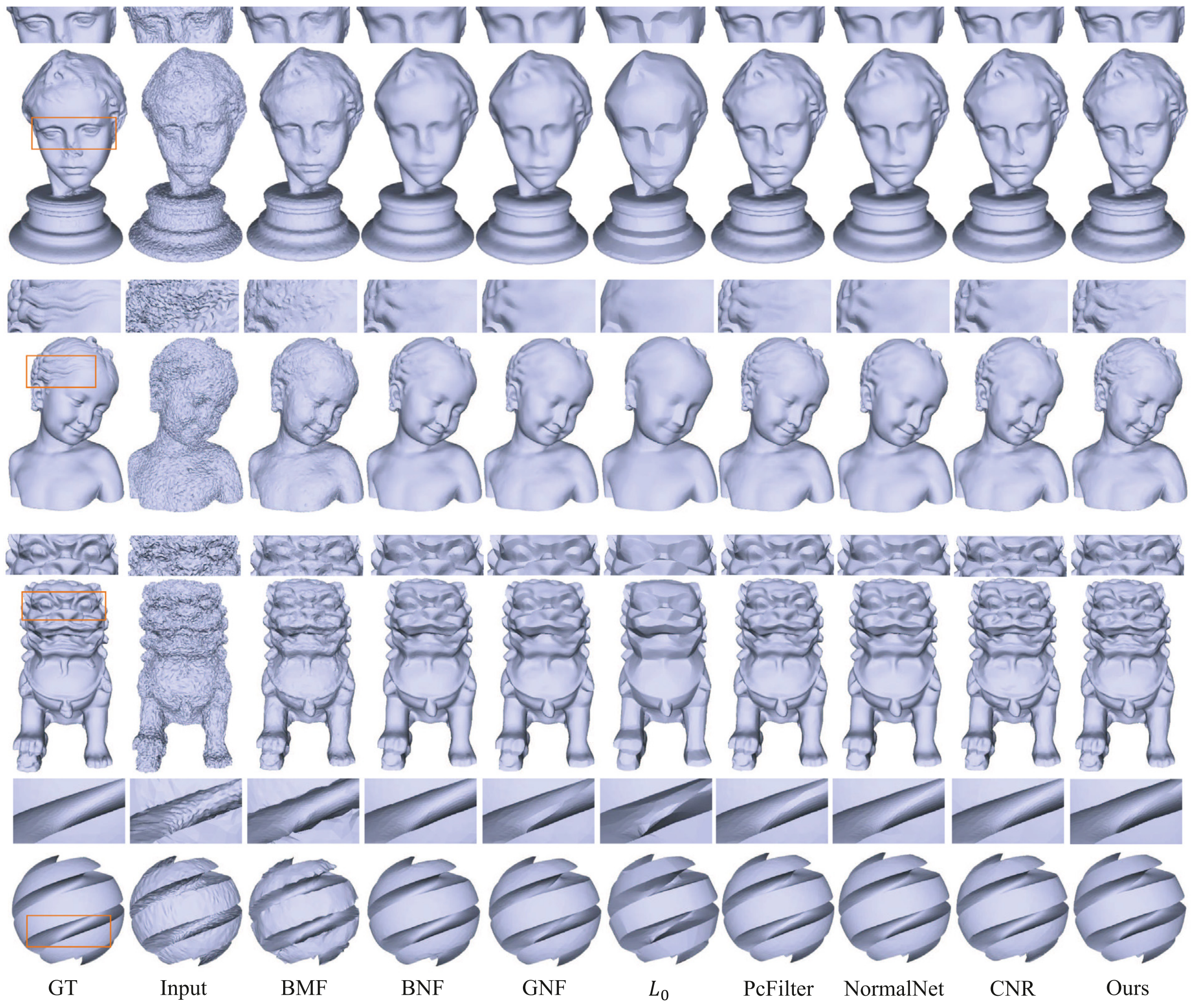


Fig. 14. Comparison of denoising approaches on Synthetic models: Eros, Child, Chineselion, and SharpSphere. By using the normal-refinement strategy, our method behaves well on the high-level noise and preserves surface features better. From the left column to the right: The ground truth; the noisy models (the Eros with Gaussian noise of $\sigma_E = 0.2$, the Child and Chineselion with Gaussian noise of $\sigma_E = 0.3$, the Sharpsphere with Gaussian noise of $\sigma_E = 0.1$), and the denoising results from BMF [15] with parameters $(n = 12)$ $(n = 10)$ $(n = 10)$ $(n = 15)$, BNF [16] $(\sigma_s = 0.45, n_1 = 20, n_2 = 20)$ $(\sigma_s = 0.35, n_1 = 20, n_2 = 10)$ $(\sigma_s = 0.35, n_1 = 20, n_2 = 10)$ $(\sigma_s = 0.35, n_1 = 10, n_2 = 10)$, GNF [17] $(\sigma_s = 0.45, n_1 = 20, n_2 = 15)$ $(\sigma_s = 0.35, n_1 = 20, n_2 = 10)$ $(\sigma_s = 0.35, n_1 = 20, n_2 = 10)$ $(\sigma_s = 0.35, n_1 = 20, n_2 = 10)$, L_0 minimization [18] $(\mu = \sqrt{2}, \alpha = 0.1\overline{\gamma}, \lambda = 0.02\ell_e^2\overline{\gamma})$, PcFilter [19] $(\sigma_s = 0.3, n_1 = 8, n_2 = 10)$ $(\sigma_s = 0.35, n_1 = 10, n_2 = 10)$ $(\sigma_s = 0.4, n_1 = 20, n_2 = 20)$ $(\sigma_s = 0.3, n_1 = 5, n_2 = 8)$, Normal-Net [20], CNR [13], and our NormalF-Net.

References

[1] Wang J, Gu D, Yu Z, Tan C, Zhou L. A framework for 3D model reconstruction in reverse engineering. Comput Ind Eng 2012;63(4):1189–200.
[2] Chidambaram S, Zhang Y, Sundararajan V, Elmqvist N, Ramani K. Shape structuralizer: Design, fabrication, and user-driven iterative refinement of 3D mesh models. In: Proceedings of the 2019 CHI conference on human factors in computing systems. 2019. p. 663.
[3] Wu J, Wang CCL, Zhang X, Westermann R. Self-supporting rhombic infill structures for additive manufacturing. Comput Aided Des 2016;80:32–42.
[4] Dong S, Xu K, Zhou Q, Tagliasacchi A, Xin S, Nießner M, et al. Multi-robot collaborative dense scene reconstruction. ACM Trans Graph 2019;38(4):84:1–84:16.
[5] Yang M, Ye J, Ding F, Zhang Y, Yan D. A semi-explicit surface tracking mechanism for multi-phase immiscible liquids. IEEE Trans Vis Comput Graph 2019;25(10):2873–85.
[6] Wang J, Yu Z. A novel method for surface mesh smoothing: Applications in biomedical modeling. In: Proceedings of the 18th international meshing roundtable. 2009. p. 195–210.
[7] Wei M, Zhu L, Yu J, Wang J, Pang W, Wu J, et al. Morphology-preserving smoothing on polygonized isosurfaces of inhomogeneous binary volumes. Comput Aided Des 2015;58:92–8.
[8] Zhang Y, Hughes TJR, Bajaj CL. Automatic 3D mesh generation for a domain with multiple materials. In: Proceedings of the 16th international meshing roundtable. 2007. p. 367–86.
[9] Huang Z, Zou M, Carr N, Ju T. Topology-controlled reconstruction of multi-labelled domains from cross-sections. ACM Trans Graph 2017;36(4):76:1–76:12.
[10] Huang Q, Flöry S, Gelfand N, Hofer M, Pottmann H. Reassembling fractured objects by geometric matching. ACM Trans Graph 2006;25(3):569–78.
[11] Yi C, Zhang Y, Wu Q, Xu Y, Remil O, Wei M, et al. Urban building reconstruction from raw LiDAR point data. Comput Aided Des 2017;93:1–14.
[12] Wang J, Xu K. Shape detection from raw LiDAR data with subspace modeling. IEEE Trans Vis Comput Graphics 2017;23(9):2137–50.
[13] Wang P, Liu Y, Tong X. Mesh denoising via cascaded normal regression. ACM Trans Graph 2016;35(6):232:1–232:12.
[14] Wei M, Liang L, Pang W, Wang J, Li W, Wu H. Tensor voting guided mesh denoising. IEEE Trans Autom Sci Eng 2017;14(2):931–45.
[15] Fleishman S, Drori I, Cohen-Or D. Bilateral mesh denoising. In: Proceedings of SIGGRAPH. 2003. p. 950–3.
[16] Zheng Y, Fu H, Au OK-C, Tai C-L. Bilateral normal filtering for mesh denoising. IEEE Trans Vis Comput Graphics 2011;17(10):1521–30.
[17] Zhang W, Deng B, Zhang J, Bouaziz S, Liu L. Guided mesh normal filtering. Comput Graph Forum 2015;34(Special Issue of Pacific Graphics 2015):1–12.
[18] He L, Schaefer S. Mesh denoising via l0 minimization. In: SIGGRAPH. 2013. p. 64:1–8.
[19] Wei M, Huang J, Xie X, Liu L, Wang J, Qin J. Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery. IEEE Trans Vis Comput Graph 2019;25(10):2910–26.

[20] Zhao W, Liu X, Zhao Y, Fan X, Zhao D. NormalNet: Learning based guided normal filtering for mesh denoising. 2019, CoRR abs/1903.04015.
[21] Wang J, Zhang X, Yu Z. A cascaded approach for feature-preserving surface mesh denoising. Comput Aided Des 2012;44(7):597–610.
[22] Wang P, Fu X, Liu Y, Tong X, Liu S, Guo B. Rolling guidance normal filter for geometric processing. ACM Trans Graph 2015;34(6):173:1–9.
[23] Zhang J, Deng B, Hong Y, Peng Y, Qin W, Liu L. Static/dynamic filtering for mesh geometry. IEEE Trans Vis Comput Graph 2019;25(4):1774–87.
[24] Wang R, Yang Z, Liu L, Deng J, Chen F. Decoupling noise and features via weighted l1-analysis compressed sensing. ACM Trans Graph 2014;33 (2):18.
[25] Zhang H, Wu C, Zhang J, Deng J. Variational mesh denoising using total variation and piecewise constant function space. IEEE Trans Vis Comput Graph 2015;21(7):873–86.
[26] Wang J, Huang J, Wang FL, Wei M, Xie H, Qin J. Data-driven geometry-recovering mesh denoising. Comput Aided Des 2019;114:133–42.
[27] Wei M, Guo X, Huang J, Xie H, Zong H, Kwan R, et al. Mesh defiltering via cascaded geometry recovery. Comput Graph Forum 2019;38(7):591–605.
[28] Li X, Zhu L, Fu C-W, Heng P-A. Non local low rank normal filtering for mesh denoising. Comput Graph Forum 2018;37:155–66.
[29] Wang P, Sun C, Liu Y, Tong X. Adaptive O-CNN: a patch-based deep representation of 3D shapes. ACM Trans Graph 2018;37(6):217:1–217:11.
[30] Jones TR, Durand F, Desbrun M. Non-iterative, feature-preserving mesh smoothing. ACM Trans Graph 2003;22(3):943–9.
[31] Yagou H, Ohtake Y, Belyaev A. Mesh smoothing via mean and median filtering applied to face normals. In: Geometric modeling and processing. Theory and applications. IEEE; 2002, p. 124–31.
[32] Yagou H, Ohtake Y, Belyaev AG. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding. In: Proceedings computer graphics international 2003. IEEE; 2003, p. 28–33.
[33] Shen Y, Barner KE. Surface denoising with directional fuzzy vector median filtering. In: 2003 international conference on multimedia and expo, vol. 1. IEEE; 2003, p. I–237.
[34] Li T, Wang J, Liu H, Liu L-g. Efficient mesh denoising via robust normal filtering and alternate vertex updating. Front Inf Technol Electron Eng 2017;18(11):1828–42.
[35] Wei M, Yu J, Pang W, Wang J, Qin J, Liu L, et al. Bi-normal filtering for mesh denoising. IEEE Trans Vis Comput Graph 2015;21(1):43–55.
[36] Fan H, Yu Y, Peng Q. Robust feature-preserving mesh denoising based on consistent subneighborhoods. IEEE Trans Vis Comput Graphics 2009;16(2):312–24.
[37] Yadav SK, Reitebuch U, Polthier K. Mesh denoising based on normal voting tensor and binary optimization. IEEE Trans Vis Comput Graph 2017;24(8):2366–79.
[38] Zhao W, Liu X, Wang S, Fan X, Zhao D. Graph-based feature-preserving mesh normal filtering. IEEE Trans Vis Comput Graphics 2019.
[39] Wu X, Zheng J, Cai Y, Fu C. Mesh denoising using extended ROF model with L_1 fidelity. Comput Graph Forum 2015;34(7):35–45.
[40] Zhu L, Fu C, Jin Y, Wei M, Qin J, Heng P. Non-local sparse and low-rank regularization for structure-preserving image smoothing. Comput Graph Forum 2016;35(7):217–26.
[41] Guo Q, Gao S, Zhang X, Yin Y, Zhang C. Patch-based image inpainting via two-stage low rank approximation. IEEE Trans Vis Comput Graph 2017;99:1–13.
[42] Zhang L, Zuo W. Image restoration: From sparse and low-rank priors to deep priors [lecture notes]. IEEE Signal Process Mag 2017;34(5):172–9.
[43] Hou Y, Xu J, Liu M, Liu G, Liu L, Zhu F, et al. NLH: A blind pixel-level non-local method for real-world image denoising. 2019, arXiv preprint arXiv:1906.06834.
[44] Chen H, Huang J, Remil O, Xie H, Qin J, Guo Y, et al. Structure-guided shape-preserving mesh texture smoothing via joint low-rank matrix recovery. Comput Aided Des 2019;115:122–34.
[45] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 652–60.
[46] Roveri R, Öztireli AC, Pandele I, Gross M. Pointpronets: Consolidation of point clouds with convolutional neural networks. Comput Graph Forum 2018;37(2):87–99.
[47] Chen H, Wei M, Sun Y, Xie X, Wang J. Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint. IEEE Trans Vis Comput Graphics 2019;1. http://dx.doi.org/10.1109/TVCG.2019.2920817.
[48] Boulch A, Marlet R. Deep learning for robust normal estimation in unstructured point clouds. Comput Graph Forum 2016;35(5):281–90.
[49] Kim HS, Choi HK, Lee KH. Feature detection of triangular meshes based on tensor voting theory. Comput Aided Des 2009;41(1):47–58.
[50] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. 2015.
[51] He K, Zhang X, Ren S, Jian S. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition. 2016.
[52] Sun X, Rosin P, Martin R, Langbein F. Fast and effective feature-preserving mesh denoising. IEEE Trans Vis Comput Graphics 2007;13(5):925–38.